RECOMMENDER SYSTEM FOR RECIPES

by

SAI BHARATH GODA

B-Tech, Jawaharlal Nehru Technological University, 2012

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Daniel A. Andresen

Copyright

Abstract

Most of the e-commerce websites like Amazon, EBay, hotels, trip advisor etc. use recommender systems to recommend products to their users. Some of them use the knowledge of history/ of all users to recommend what kind of products the current user may like (Collaborative filtering) and some use the knowledge of the products which the user is interested in and make recommendations (Content based filtering). An example is Amazon which uses both kinds of techniques.. These recommendation systems can be represented in the form of a graph where the nodes are users and products and edges are between users and products. The aim of this project is to build a recommender system for recipes by using the data from allrecipes.com. Allrecipes.com is a popular website used all throughout the world to post recipes, review them and rate them. To understand the data set one needs to know how the recipes are posted and rated in allrecipes.com, whose details are given in the paper. The network of allrecipes.com consists of users, recipes and ingredients.

The aim of this research project is to extensively study about two algorithms adsorption and matrix factorization, which are evaluated on homogeneous networks and try them on the heterogeneous networks and analyze their results. This project also studies another algorithm that is used to propagate influence from one network to another network. To learn from one network and propagate the same information to another network we compute *flow* (influence of one network on another) as described in [7]. The paper introduces a variant of adsorption that takes the flow values into account and tries to make recommendations in the user-recipe and the user-ingredient networks. The results of this variant are analyzed in depth in this paper.

Table of Contents

List of Figures vi
List of Tables vii
Acknowledgements viii
Chapter 1 - Introduction 1
1.1 Structure of Data in allrecipes.com 3
1.2 Literature Review on Recommender Systems 5
1.2.1 Collaborative Filtering 5
1.2.2 Content Based Filtering 6
Chapter 2 - Data Acquisition 7
2.1 Web Crawler 7
Chapter 3 - Data Cleaning 10
Chapter 4 - Recommendation Algorithms 14
4.1 Adsorption 16
4.2 Matrix Factorization 19
4.3 Influence propagation in heterogeneous network 20
4.4 Mean Average Precision 22
Chapter 5 - Results 25
5.1 Statistics of different networks 25
5.2 Homogeneous Network Results 27
5.3 Heterogeneous Network Results 29
5.4 Inferences 31
5.3.1 User-Recipe Network 31
5.3.2 User-Ingredient Network 31
5.5 Testing 32
Chapter 6 - Conclusion and Future Work 35
6.1 Conclusion 35
6.2 Future Work 35
References Or Bibliography 36

Appendix A - Technologies 37
1. Apache Hadoop 37
1.1 MapReduce 37
2 Mahout 38
3 Pig 38
4 Apache Http Client 38
5 Google GSON API 39

List of Figures

Figure 1.1 Workflow of building a recommender system. 1
Figure 1.2 Structure of data in allrecipes.com 3
Figure 1.3 Class diagram of data in allrecipes.com 4
Figure 2.1 Workflow of extracting data 8
Figure 2.2 Review JSON file 8
Figure 2.3 Recipe JSON 9
Figure 2.4 Cook profile JSON 9
Figure 3.1 showing data cleaning steps that are taken to clean input data file 11
Figure 3.2 Recipe JSON file with ID 11
Figure 3.3 User with ID JSON file 12
Figure 4.1 Edge length between the user-recipe and user-ingredient graph 14
Figure 4.2 Neighbors and weights of neighbors are calculation 16
Figure 4.3 Formulae to calculate P^{cond} and Y^{v} 18
Figure 4.4 Adsorption algorithm 18
Figure 4.5 Adsorption workflow for user-recipe network 19
Figure 4.6 Adsorption workflow for user-ingredient network 20
Figure 4.7 Workflow for calculating flow values 22
Figure 4.8 Adsorption workflow for user-recipe network with flow values 22

List of Tables

Table 5.1 Statistics of user-recipe network for all train folds 25
Table 5.2 Statistics of user-recipe network for all test folds 26
Table 5.3 Statistics for user-ingredient network 27
Table 5.4 Adsorption results for user-recipe network 27
Table 5.5 Matrix factorization results for user-recipe network 28
Table 5.6 Adsorption results for user-ingredient network 28
Table 5.7 Matrix factorization results for user-ingredient network 29
Table 5.8 Heterogeneous algorithm results for user-ingredient network 30
Table 5.9 Heterogeneous algorithm results for user-recipe 30
Table 5.10 Comparison for all networks 32

Acknowledgements

Firstly I would like to thank the almighty for being the source of my strength. I would like to extend my sincere thanks to Dr. Doina Caragea for having trust in me and assigning this complex project. Without her help and support this project would not have been successful. I would also like to thank Dr. Daniel Andresen for being my major professor and assisting me during the course of my project. It was a pleasure to also work under Dr. Andresen for a couple of semesters as a Teaching Assistant. Thanks to him I learnt two new frameworks MVC4 and Ruby on Rails. I would like to thank Dr. Simon Ou for being a part of my committee and reviewing my thesis. I took a course called computer and information security taught by him and I thoroughly enjoyed it. I would like to thank Dr. Patrice Chalin, with whom I worked for two semesters. I really like his style of teaching and learnt a lot from. I would like to extend my sincere thanks to Dr. Gurdip Singh, for having faith in my abilities and appointing me as a TA for all the four semesters I was at Kansas State. I would also like to extend my thanks to Dr. Gustafson, who gave me the chance to study in this wonderful institution. I would also like to thank the department of Computing and Information Sciences for sponsoring my education for the past two years. I would also like to thank other professors that I have worked with namely, Dr. William Hsu, Dr. John Hatcliff, Dr. Rod Howell and Dr. Scott Delaoch. I would like to thank the CIS support staff in assisting me with any official paper work. Last but not the least, I would like to thank to my family and friends who have been my pillars of strength and been with me though the good and bad times. It's because of their encouragement and support that I was able to successfully completed my course work here at Kansas State.

Chapter 1 - Introduction

Recommender systems is an important research area in the field of computer science. Most of the big players in the industry like Google, Amazon, EBay, etc. use some kind of recommender systems in the background to recommend personalized products, ads, videos to the user. Most of the recommender systems recommend things that are relevant to the user. They make use of the user's browsing history, search history, emails, etc. to gather information. The workflow of building a recommender system is shown below:

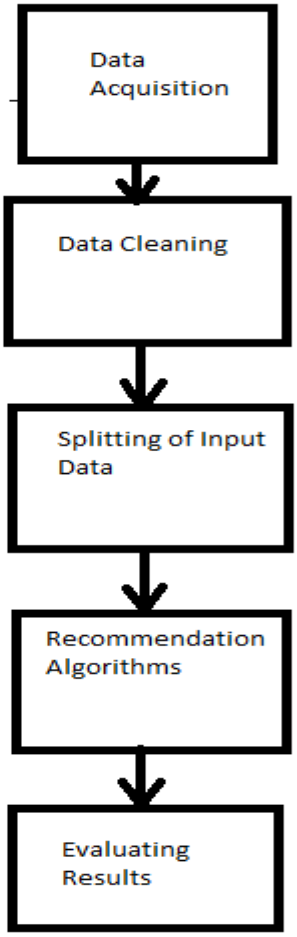


Figure 1.1 Workflow of building a recommender system.

The first step in building a recommender system is to gather data from the user. Since of these websites have personal user accounts and access to browsing and search history, they have

significant information from which they can recommend things to the user. We are looking at recommender systems that take user preferences and try to recommend something which is similar to what the user prefers. Hence we need some kind of data that has user's and their preferences. One such website where you have user accounts and their preferences is allrecipes.com. The structure of data in allrecipes.com is explained in detail in section 1.1 and data acquisition is explained in chapter 2.

The next step in building a recommender system is to clean the data that we have and generate input in acceptable formats for the algorithms to run effectively. Data cleaning includes removal of stop words(most commonly used words like a, an, the... which have no significant bearing), special symbols and stemming of the words (stemming is removal of common prefixes and suffixes in words). There might be words in our data do not belong to the English language. I have only considered those recipes and ingredients that are posted in allrecipes.com' United States website. Hence English is my primary language that I focus on. So I remove all the other words that do not belong to the English language. Data cleaning is explained in detail in chapter 3.

The next step in building a recommender system is dividing your data into two parts -- training data and test data. The training data is given as an input to the algorithm and results are compared with the test data. In order to test the accuracy of the algorithm we make use of random splitting and this is done five times to generate 5 different train and test folds.

The next step in building a recommender system is running the algorithms on the input data. There are several algorithms that I have used in my research like adsorption, matrix factorization, KNN, BFS and link prediction in heterogeneous networks. All the algorithms and work flows are explained in detail in chapter 4.

The next step in building a recommender system is evaluating your results. The results are evaluated against what we have in the test data set. The main evaluation metric that I have used in my research is Mean Average Precision. The evaluation part is explained in detail in chapter 4.

All my results are compared in with the matrix factorization algorithm that was used in the Netflix challenge. The results are explained in detail in chapter 5. The different technologies that I have used in this project are explained in the technology section (1.2) of chapter 1. The basic workflow in building a recommender system is:

1.1 Structure of Data in allrecipes.com

Allrecipes.com is a prominent website used in United States and all throughout the world by many users to post recipes, review recipes posted by others and rate recipes posted by other users. Users of allrecipes.com have to create an account to be able to review and rate recipes. They have the option of adding recipes they like to the **recipe box**. Typically a user doesn't have any constraint on how many recipes he can add to the recipe box. A user can only rate and review a recipe that is present in a recipe box. He cannot rate and review recipes that are outside his box. General tendency is that whenever a user wants to try out a new recipe or a recipe he likes he adds it to his recipe box. A typical data model consisting of user, recipe and ingredients is given below:

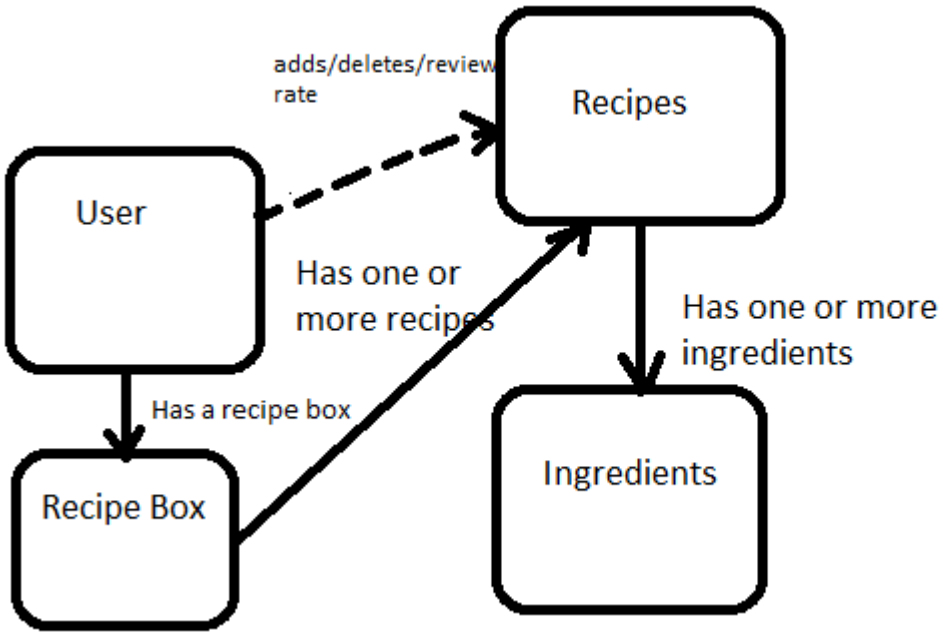


Figure 1.2 Structure of data in allrecipes.com

A recipe consists of one or more ingredients. The ingredients and the quantity(Ex:- pinch of salt) to be used is specified for each recipe in allrecipes.com. The amount of ingredient used in the recipe is ignored for my experiments but can be included in future scope of this project. The

recipe is also categorized in the website according to the cuisine. This is not used in my current experiments but can be included in the future scope of this project where we could consider only recipes in one type of cuisine and make recommendations in that cuisine . The only data that we used in the website are the ingredients, recipes and user profiles. Using these we constructed user, ingredient, recipe network and then implemented some algorithms on them.

The class diagram of recipe, ingredient and user is given below:

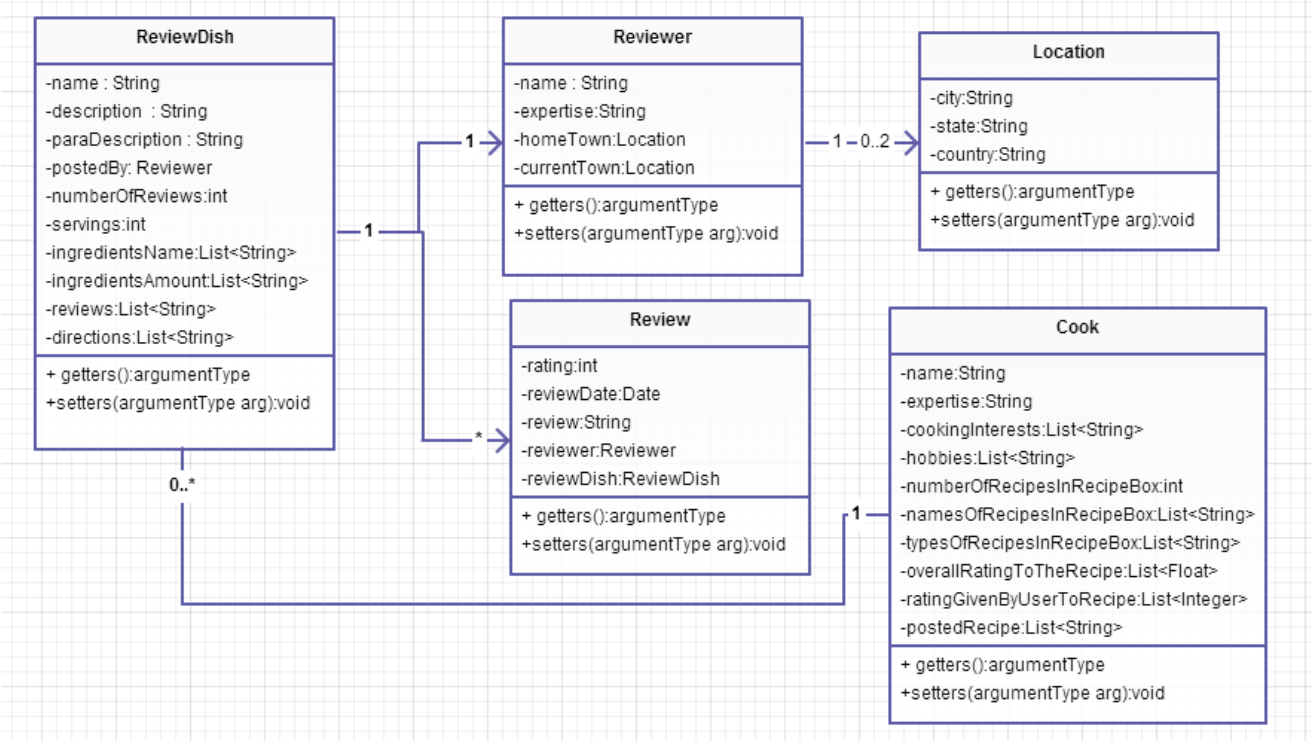


Figure 1.3 Class diagram of data in allrecipes.com

The data extracted from the website is filled into these objects which are in turn converted to JSON strings and stored in text files and processed using hadoop. The process of extracting data is explained in chapter 2 and technologies used are explained in the appendix section.

1.2 Literature Review on Recommender Systems

Recommender systems try to predict the 'preference' that the user would give to an 'item'. Recommender systems used by many e-commerce, travel, retail, movie, music, books etc. applications. The two main approaches of building a recommender system are

1. Collaborative filtering
2. Content based filtering

1.2.1 Collaborative Filtering

Collaborative filtering method uses the user's preferences like the rating, review or behavior to predict what they might like based on their similarity to other users. Collaborative filtering algorithms are independent of the domain, which is their biggest advantage when compared to content based algorithms. There are many approaches are used to predict the similarity between the two users. Some of them are K-nearest neighbors, Pearson Correlation etc.

User profile data is often available in two forms. One is implicit data where the system analyzes what items the user has viewed, what purchases user has made or analyzing user's social network to understand users preferences. Other is explicit data where the user is asked to review an item, rate an item, rank a collection or create a wish list.

Any collaborative filtering system mainly uses two steps to generate recommendations.

1. The current user is compared against all other users in the system and similarity is calculated.
2. Recommendations are made to the current user based on the items liked by 'like-minded users' (similar users).

The metric used for computing similarity in the experiments is cosine similarity which is explained in detail in chapter 4. The algorithms that are discussed in the later chapters like adsorption[5] and matrix factorization[6] are collaborative algorithms that use ratings and users preferences and make recommendations.. These collaborative filtering algorithms are analyzed in great detail in chapter 4.

Collaborative filtering suffers from 'cold start' problem. Whenever new ratings and reviews are posted, the similarity between two users changes and hence the recommendations

made also change. Hence the system needs to be constantly updated. It also suffers from sparsity in data. Most of the items in the system are not rated by the user. Accuracy of the results decreases because similarity between users changes

1.2.2 Content Based Filtering

Content based filtering methods are based on the item description rather than user's explicit feedback like ratings or reviews. They are also dependent on user's preferences specified in his profile (things that the user likes, his hobbies, interests to make recommendations). Content based filtering algorithms store keywords to indicate what kind of items user likes (For example, if the user likes 'Conjuring movie' then you associate the keyword 'horror films' to this user).

The content based filtering algorithms take into account things that user liked in the past (based on his profile and also on the type of items that he liked) and try to recommend something similar. Various items are compared against the user's preferences and the top results are recommended.

Based on the user's preferences a weighted vector is created for an item which is compared against other items and the top results are recommended. The weights in this vector can be computed using a variety of techniques like Bayesian Classifiers, cluster analysis, decision trees or artificial neural networks.

Chapter 2 - Data Acquisition

Getting the data of recipes, reviews and cooks was a challenging problem because most websites provide RESTful services that applications can use to get data. But there were no RESTful API's that are provided by allrecipes.com. Hence I wrote a *web crawler* to extract data from allrecipes.com. The description of a web crawler is given below:

2.1 Web Crawler

A web crawler is a computer program that scans the world wide web and gathers information. Generally big search engines like Google, Ask, Yahoo use some kind of web crawlers to repeatedly crawl the web and extract data. The crawling policy of each web site is specified in robots.txt.

So I have checked the robots.txt of allrecipes.com and made sure I adhered to all the rules specified in the robots file. I have crawled the recipe, review and cook profiles using Apache Http Client API. I used xml parser called JSOUP to parse the HTML page that is returned by the Http Client. From the HTML page I extracted what details I wanted and put them into model objects described in the above class diagram. Later after extracting all details I need I have stored these objects as JSON strings in text files which can be used for processing by hadoop.

The workflow of extracting the data from the website is explained in the following steps:

1. The crawler bot sends request to the web server of allrecipes.com asking for the recipe/cook/review page.
2. The server of allrecipes.com returns a web page which is in the form of HTML to the bot.
3. The bot then passes this HTML file to the parser program to extract information from this HTML page. This parser fills in the model objects.
4. The model objects are converted to JSON strings and JSON strings are stored in HTML file.

The figure showing the above workflow is given below**.**

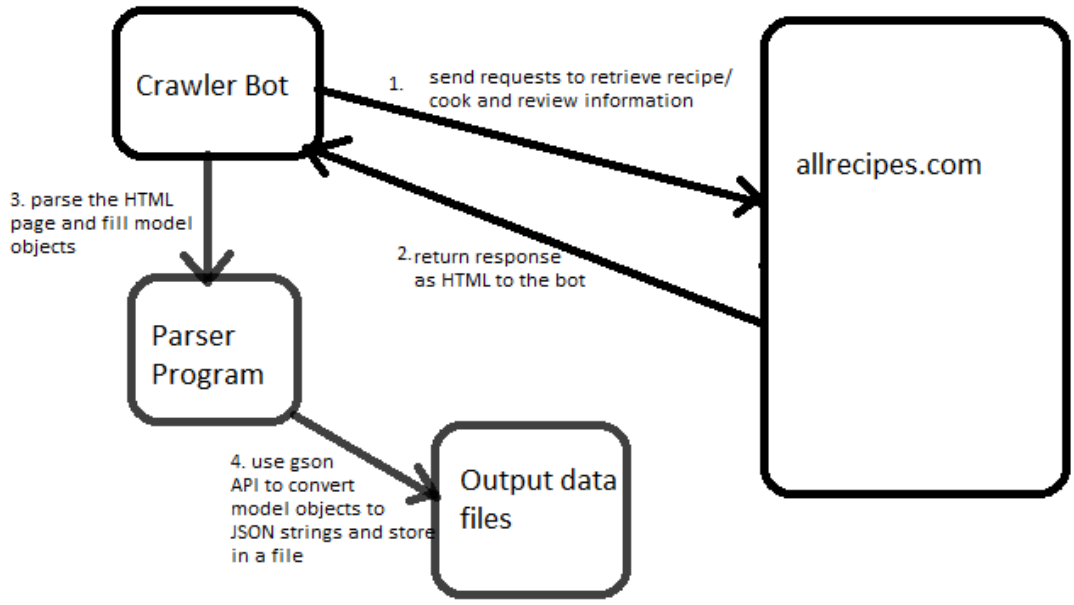
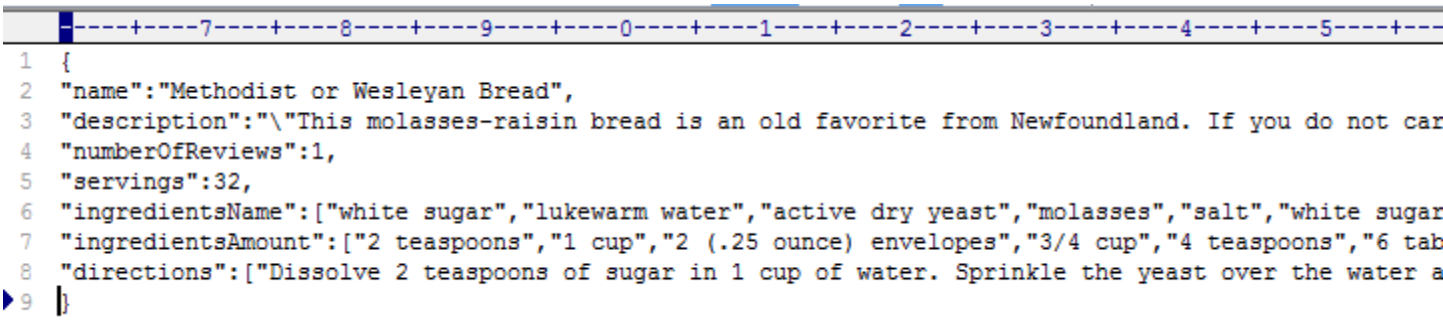


Figure 2.1 Workflow of extracting data

The preview of JSON data is shown in figures below:

Figure 2.2 Review JSON file


```
{
"rating":3,
"reviewDate":"Feb 23, 2011 12:00:00 AM",
"review":"These turned out quite dry. I made these as mini-muffins...so maybe that was the issue.
"reviewer":{"name":"chickysmama","expertise":" Expert","homeTown":{"city":"Aberdeen","state":" Sou
"reviewDish":{"name":"To Die For Blueberry Muffins","paraDescription":"Extra big blueberry muffins
}
```



```
{
"name":"Methodist or Wesleyan Bread",
"description":"\"This molasses-raisin bread is an old favorite from Newfoundland. If you do not car
"numberOfReviews":1,
"servings":32,
"ingredientsName":["white sugar","lukewarm water","active dry yeast","molasses","salt","white sugar
"ingredientsAmount":["2 teaspoons","1 cup","2 (.25 ounce) envelopes","3/4 cup","4 teaspoons","6 tab
"directions":["Dissolve 2 teaspoons of sugar in 1 cup of water. Sprinkle the yeast over the water a
}
```

Figure 2.3 Recipe JSON


```
{
"name":"Janice",
"expertise":" Intermediate",
"cookingInterests":["Grilling \u0026 BBQ","Stir Frying","Asian","Mexican","Indian","Italian","Southern
Eastern","Mediterranean","Healthy","Vegetarian","Dessert","Quick \u0026 Easy","Gourmet"],
"hobbies":["Gardening","Hiking/Camping","Walking","Reading Books","Music"],
"numberOfRecipesInRecipeBox":1,
"namesOfRecipesInRecipeBox":["Lemon-Scented Tuscan Ravioli with Spinach and Raisins"],
"typesOfRecipesInRecipeBox":["Personal Recipe"],
"overallRatingToTheRecipe":[0.0],
"ratingGivenByUserToRecipe":[0]
}
```


Figure 2.4 Cook profile JSON

There are three different files that store the review, recipe and cook profile data. These files store about 2 million reviews data, 60k recipes data and 20k user data respectively. This data is cleaned in the next step. The process of cleaning the data is explained in detail in chapter 3. After cleaning the data the same data is regenerated but in a clean form. The cleaned data set is then given as an input to these algorithms that work on that data to produce results.

The server of allrecipes.com blocks IP addresses that send more than 3 requests per second. Hence fewer requests were sent to the server. I used Beocat cluster to submit a job that ran for 80 hours to crawl the entire data of recipes. Reviews and cook profiles took a lot more time to crawl because more requests had to be sent to the server. Overall almost entire data of allrecipes.com has been crawled.

Chapter 3 - Data Cleaning

Data cleaning is a process of removing unnecessary symbols, trimming unnecessary prefixes and suffixes and also removing commonly used words that have no special meaning (For example words like a, an, the which , when, why, what, are, for.. etc do not have any special meaning when it comes to explaining something about the domain). Such words are call *stop words***.** A collection of such words is made and these all the words are removed from the data. We only clean ingredient data because recipe data is clean and cook profiles have unique usernames and hence the only thing that had be cleaned was the ingredients. Many ingredients are the same but the way of representing them by different users is different. For example salt is a common ingredient that is used by many users. Some users specify salt as pinch of salt, some as tea spoon of salt and others specify only salt. As a human we can say that all are the same but for a computer to be able to do that we must remove words like pinch, tea-spoon to tell the computer that they are the same.

There are many steps that are followed by different researchers to clean their data. The steps that I followed to clean the data are:

1. The first step is to trim the leading and trailing spaces in my text.
2. The second step is to remove unnecessary symbols other that the alphabets (lower and upper case) and numbers.
3. The third step is to remove the 'stop words' from the text.
4. The fourth step is to split the input phrase into set of words and stem each word using an English Stemmer. A stemmer is a computer program that removes unnecessary prefixes and suffixes in a word.
5. The next step is to remove one letter words that remain in the output document.
6. The other remaining words are the clean words that are domain specific and are useful for our experiments.
7. Such words are given unique ID's in the next step.
8. The words in the original data set are replaced with the ID's that we assigned in - the previous step.
9. Now the input file contains ID's instead of raw text.

The diagram showing how input text is cleaned and how a new file is generated for the algorithm is shown below:

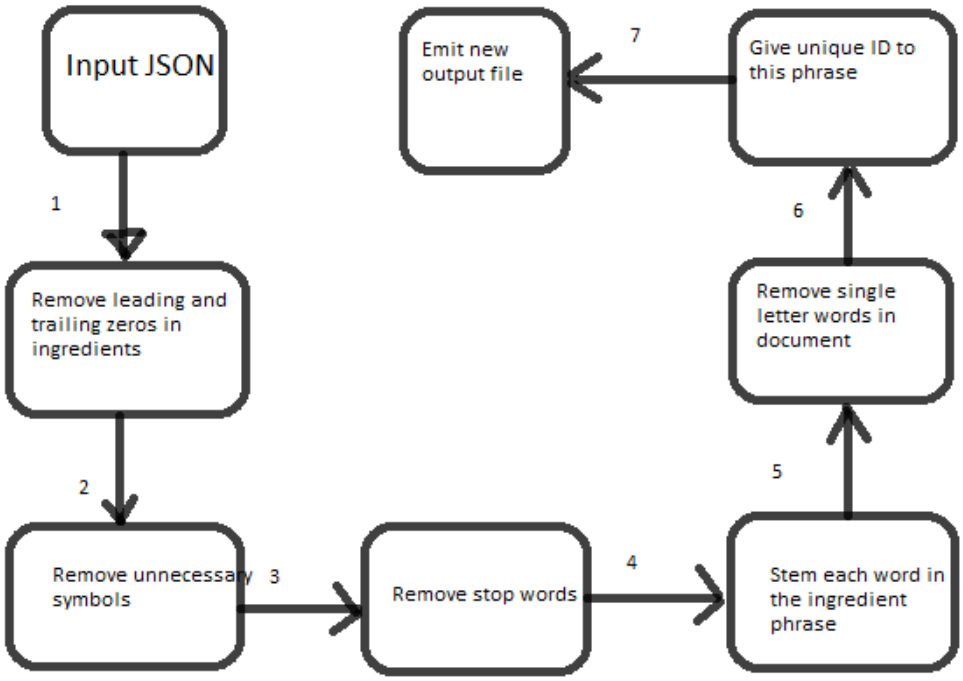


Figure 3.1 showing data cleaning steps that are taken to clean input data file

The advantage of using an ID over raw text is that IDs are unique and there is no point of checking for redundancy in every experiment that we perform, data is consistent and also reliable. The figures showing the updated JSON's are given below:


```
{"name":"10000_recipe",
"description":"\"This great frozen dessert will hold in the freezer for weeks. Everyone will love it! 
"postedBy":{"name":"9732_user"},
"numberOfReviews":47,"servings":9,
"ingredientsName":["3731_ingredient","13888_ingredient","9403_ingredient","12243_ingredient","615_ingr
"ingredientsAmount":["1 (8 ounce) package","1 cup","1 (15 ounce) can","1 (16 ounce) container","1 (10 
"directions":["In a large bowl, cream together the cream cheese and sugar. Fold in the pineapple, whip
walnuts.","Freeze for 4 to 6 hours or overnight. Remove from freezer 1 to 2 hours before serving. Enjo
```


Figure 3.2 Recipe JSON file with ID

```
{"name":"9_user",
"expertise":" Not Rated",
"numberOfRecipesInRecipeBox":79,
"namesOfRecipesInRecipeBox":["9684_recipe","23264_recipe","55152_recipe","9041_recipe","35308_recipe","
_recipe","59493_recipe","1020_recipe","22321_recipe","16466_recipe","46985_recipe","29538_recipe","3651
ipe","1555_recipe","35270_recipe","16440_recipe","14392_recipe","19204_recipe","15715_recipe","2578_rec
"Swanson(R) Black Bean, Corn and Turkey
Chili","43243_recipe","14504_recipe","8222_recipe","53725_recipe","46623_recipe","12898_recipe","59155_
Strawberry
Trifle","38077_recipe","33332_recipe","12293_recipe","47836_recipe","52255_recipe","59513_recipe","1050
 Cookie
Bark","13886_recipe","8998_recipe","15834_recipe","19536_recipe","29407_recipe","47757_recipe","45394_r
","37010_recipe","35772_recipe","4429_recipe","31207_recipe","56990_recipe","7216_recipe","35305_recipe
6161_recipe","14685_recipe"],
"typesOfRecipesInRecipeBox":["Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Personal Recipe"
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved","Kitchen Approved
"overallRatingToTheRecipe":[4.78,4.9,4.46,4.28,4.55,4.67,4.48,4.54,4.71,4.59,4.25,3.77,4.82,4.66,4.59,4
4,4.25,4.73,4.55,4.72,4.49,4.59,4.72,4.77,4.62,4.75,4.58,4.49,4.17,4.62,0.0,4.62,4.71,4.49,4.74,4.66,4.
.67,4.8,4.49,4.75,4.56,4.33,4.79,4.55,4.3,4.71,4.67,4.31,4.53,4.78,4.58,4.47,4.82],
"ratingGivenByUserToRecipe":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,5,0,0,0,0,
0,0,0,0,0,0,0,0,0,4,0],
"postedRecipe":["53186_recipe"]}
```


Figure 3.3 User with ID JSON file

Only 'kitchen approved' recipes are taken into consideration and all the kitchen approved recipes in the data set have unique ID's. Only these recipes are taken into consideration because other recipes are posted by different users but allrecipes.com has not approved it yet or the recipe is rejected after tested by the experts al allrecipes.com. Kitchen approved recipes are available public and all users of the website are able to see it. Personal recipes do not have public facing and receive very less or no views at all. These are generally because recipes of such type have already been posted before or they have not been approved. So among the data set we filter the kitchen approved recipes only and send these to the algorithms. The next step is running different algorithms for which some values need to be calculated which is explained in detail in chapter 4.

Some statistics of the data are:

- Total number of recipes = 59805
- Total number of users = 20268
- Total number of ingredients = 14242

Chapter 4 - Recommendation Algorithms

There are several recommendation algorithms that have been used in my experiments and even combination of algorithms have been used. The major algorithms that I have worked with are adsorption, matrix factorization ALS with implicit and explicit feedback and link prediction in heterogeneous network using flow equation(influence propagation). For all these algorithms we must transform the JSON files into specific formats for this algorithms to work effectively. The main steps include graph construction, generating distribution, finding weights of nodes in the graph and finding neighborhood. Before all this is done the input data is divided into train and test data. The steps involved in preparing the graph, building neighborhood is shown below:

1) From the recipe JSON file shown in figure 3.2 and 3.3 we generate user-ingredient count. For example in figure 3.3 we have 9_user who is associated with many recipes(79 to be precise). Among these recipes we pick the '*kitchen approved*' recipes and see if the rating given to the kitchen approved recipe by the user is 4 or greater. For all such recipes we take ingredients specified in step 2 and emit (9_user,4567_ingredient) and 1 (where 1 is the count of user-ingredient pair). For all such recipes we emit the counts. we generate a CSV file that contains the user - ingredient count.
2) The same step is repeated for the data in figure 3.3 where in all "*kitchen approved*" recipes are emitted along with their rating. For example in figure 3.3 we emit 9_user,9684_recipe,1. The '1' here is determined as given in the figure below:

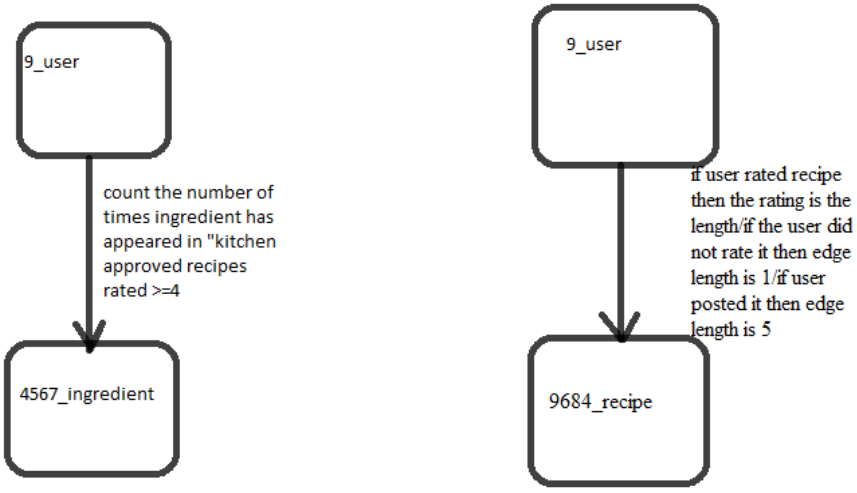


Figure 4.1 Edge length between the user-recipe and user-ingredient graph

3) Now that we have built the graph as shown in the figure the next step is to divide the data into train-test folds. The minimum condition for dividing the data into train and test folds is that a user should have at least two edges. All users with one or zero edges are removed in this step. The data is divided in the ratio of 70-30 percent in random fashion. Five such folds are created for user-recipe and user-ingredient graphs. We only tamper with the train folds but not the test folds. The test folds remain as they are throughout all the experiments.
4) For each fold that we have we calculate a metric called 'distribution'. The distribution for a user is defined by the formula user-item count(where item is either recipe or ingredient depending on the graph we choose)/sum of counts of all items. We calculate distribution for each edge in the graph.
5) Using the distribution file that we emitted in the previous step we calculate the degree of the user. Degree of the node is defined by the number of edges coming out of the node. The output of this step is user: degree item1,item2,item3....
6) The next step in preparing the input for the algorithms is to calculate the neighborhood of the user. The neighborhood of the user is calculated in two steps:
    a) For every item in the output of step 5 we emit item and user: degree. For each item we create a file that stores the item name and the users that have the item.
    b) From the output in step 4 we take a user and an associated item. Then we look up for the item file and add the list of users of that item to the neighborhood. The weight will be 1 and will be incremented if there are more than one items in common in between users. Finally after this step we emit user: degree user1:degree1:weight1, user2:degree2:weight2.... The neighborhood step is better explained in the diagram below:

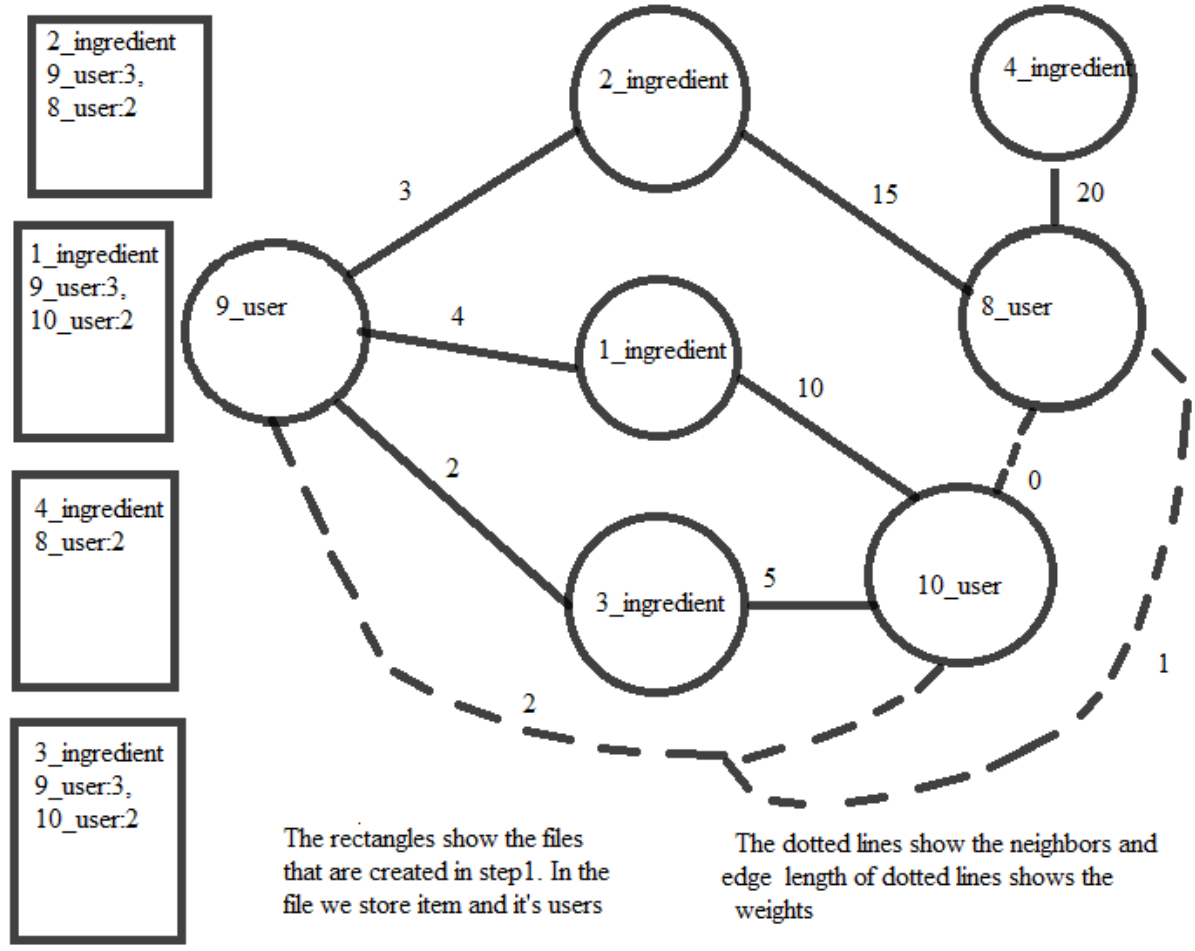


Figure 4.2 Neighbors and weights of neighbors are calculation

The algorithm described above is called Nearest Neighbor algorithm and a variant of it is called KNN, K-Nearest Neighbors. This is the first algorithm that is used in my experiments. The neighborhood is taken into consideration in all my experiments. The first algorithm that takes neighborhood and makes recommendations is adsorption and the other algorithm that is provided by the mahout framework is matrix factorization (alternating least squares) with implicit feedback. Each of these algorithms are described below:

4.1 Adsorption

There are three kinds of ways knowledge can be learnt or something can be inferred from the data. They are supervised learning, unsupervised learning and semi-supervised learning.

Supervised learning algorithms have labeled data and try to infer based on the things learnt from the label data. Unsupervised learning does not have labeled data and such algorithms cluster group of data and try to infer from the cluster. Semi-supervised learning algorithms make use of labeled data and unlabeled data The semi-supervised algorithms have two steps in which the first step is that the labeled data determine what they think about group of unlabeled data, and from this discussion they learn about unlabeled data.

Adsorption is a semi-supervised algorithm that is used on large scale graph data. The goal of adsorption is to label unlabeled examples and from the limited amount of knowledge they have from the labeled data. In our example, we have a user-user neighborhood. Between two users there are a few ingredients or recipes in common. We try to learn from other user's neighborhood and try to learn whether the user likes the recipes or ingredients liked by other users or not. The neighborhood and the distribution we have is labeled data and trying to determine links between users using the above data is unlabeled data and from the labeled data we are trying to infer something about unlabeled data. Adsorption is one such algorithm that works on semi supervised graph data.

Adsorption propagates label-information from the labeled data to other data via edges. Given an undirected graph G(V,E,W) and a vertex v, and edge (a, b), where a, b belong to V and W(a,b) belongs to R^+ where R is the strength of the similarity, such a graph is sent as an input to adsorption.

The number of vertices in the graph are n, where n=|V| and n_l is the number of labeled vertices and n_u is the number of unlabeled vertices. $n_u + n_l = n$. Let 'L' be the set of labels and m=|L|. Each vertex v is associated with a row vectors Y_v and $Y^{\wedge}_v$. The lth element of vector Y_v indicates prior knowledge and a higher value of Y_{vl} indicates that the vertex v should have label l and if $Y_{vl}=0$ then it indicates that the vertex is not associated with the label. The algorithm computes $Y^{\wedge}_v$ which has similar semantics as Y_v.

To label any vertex v we start a random-walk at vertex v with three possible actions: inject, continue and abandon. Sum of these probabilities is 1 ($p^{inj}_v + p^{cont}_v + p^{abnd}_v = 1$).. We randomly walk through the graph in three stages. We randomly walk through the graph with probability p^{inj}_v and return Y_v. We take $p^{inj}_v=0$ for unlabeled vertices. The second kind of random walk through the graph is done using p^{abnd}_v where we abandon labeling process and return a zero vector. Thirdly the random walk continues with probability p^{cont}_v to a neighbor of v, v' with a

probability of $W_{v,v'}$ >=0. The weight W(v,v') is 0 if (v,v') do not belong to edge set E. the absorption algorithm and Pr[v'/v] calculation is shown below:

$Pr[v'/v] = w_{v'v} / \Sigma W_{uv}$ (for all u and (u,v) $\in$ E) if (v',v) $\in$ E or .....(1)

$Pr[v'/v] = 0$

The expected score $Y^{\wedge}_v$ is given by

$Y^{\wedge}_v = p^{inj}_v \times Y_v + p^{cont}_v \times \Sigma Pr[v'/v] Y^{\wedge}_{v'}$ (for (v,v') $\in$ E) $+ p^{abnd}_v \times 0_m$ .....(2)

Figure 4.3 Formulae to calculate P^{cond} and Y^v

The below figure explains adsorption algorithm[4]:

Input:

-Graph: G = (V,E,W)

-Prior labeling : $Y_v \in R^{m+1}$ for v $\in$ V

-Probabilities: p^{inj}_v, p^{cont}_v, p^{abnd}_v for v $\in$ V

Output:

-Label Scores: $Y^{\wedge}_v \leftarrow Y_v$ for v $\in$ V

repeat:

$D_v \leftarrow \Sigma W_{uv} Y^{\wedge}_u / \Sigma W_{uv}$ for v $\in$ V

for all v $\in$ V do

$Y^{\wedge}_v \leftarrow p^{inj}_v \times Y_v + p^{cont}_v \times D_v + p^{abnd}_v \times r$

end for

until convergence

Figure 4.4 Adsorption algorithm

The algorithm shown in the above figure represents the adsorption algorithm. After running this algorithm on the above data set we get recommendations. For the user-recipe network we get recipes as recommendations among which top ten are chosen. The top ten recipes are picked based on their $Y^{\wedge}$ values and then MAP precision is run on the output of this algorithm

and the user-recipe train data. The results are shown in the chapter 5 and workflow is shown below

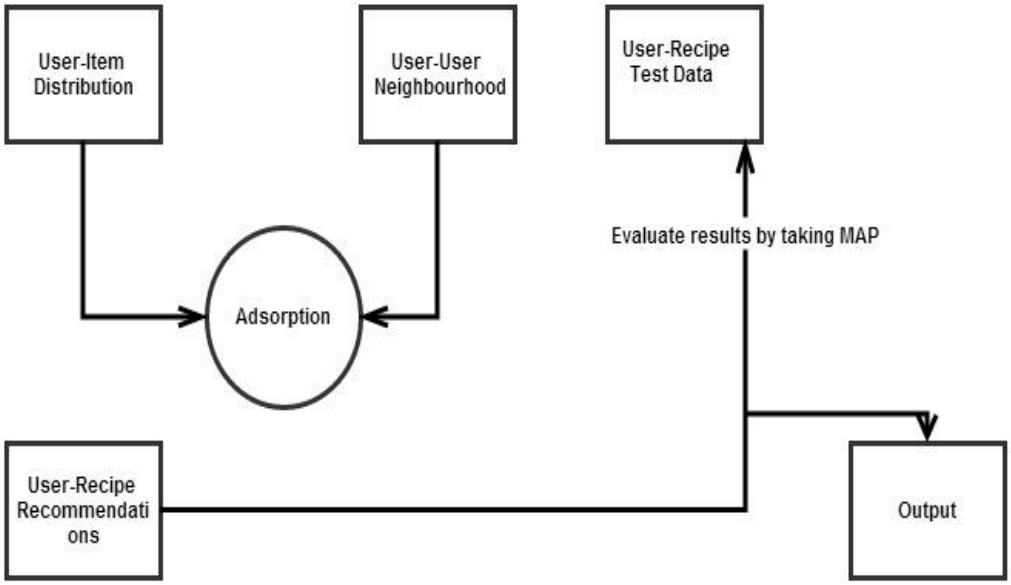


Figure 4.5 Adsorption workflow for user-recipe network

4.2 Matrix Factorization

Matrix factorization with alternating least squares having implicit and explicit feedback is provided by mahout framework. The algorithm proposed by [5] is built in the framework. T

he basic matrix factorization algorithm takes N and M which are number of user and items respectively (u as index for users and i , j as index for items) and ratings of the user (denoted by r_{ui}) and tries to predict $\hat{r}_{ui}$, which are the recommended ratings. There are two other vectors that are defined which are q_i and p_u. The elements of q_i indicate whether the item possesses those factors. The p_u vector indicated to what extent the user has interest in those items(N and M are stored in these vertices respectively). The dot product $q_i^T p_u$ captures the interest in item by a user[6].

$\hat{r}_{ui} = q_i^T p_u$.

The matrix $\hat{r}_{ui}$ contains the recommendations made for the user. This is a basic matrix factorization technique. The system regularizes squared error on a set of known ratings as:

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

The mahout framework provides this algorithm and input data is sent to this algorithm and recommendations are returned we calculate the MAP (Mean Average Precision) between the recommended data and the test data. The results are shown in chapter 5 For the user-recipe network the ALS algorithm returns a set of recipes. For the ingredient network the ALS algorithm returns set of ingredients. All ingredients are taken and cosine similarity is calculated between the output ingredients and all recipes. The top results are then taken and MAP is calculated between the returned results and the test data. The results of this is shown in detail in chapter 5 The adsorption algorithm workflow for ingredient network is shown below(The adsorption module can be replaced with matrix factorization in both cases):

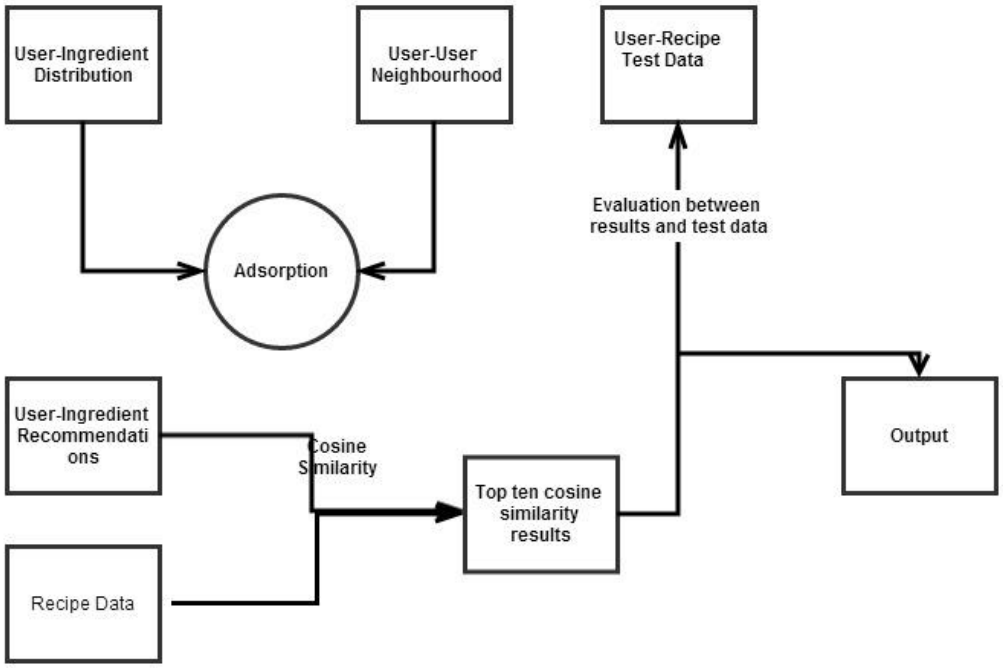


Figure 4.6 Adsorption workflow for user-ingredient network

4.3 Influence propagation in heterogeneous network

The above two algorithms, adsorption and matrix factorization are algorithms that are run on homogenous network separately. It would be good if we could learn from one network and try

to infer things in other network. There is an algorithm that is proposed in the paper "Link Prediction in Heterogeneous Network" that introduces `flow` to propagate influence from one network to other network. Flow is computed using the formula

$$flow(v,u,i) = score(v) \cdot \beta \cdot \frac{weight(v,u,i)}{degree(v,i)}$$
$$+ score(v) \cdot \beta \cdot \sum_{j \neq i}^{K} (\sigma(i,j) \cdot \frac{weight(v,u,j)}{degree(v,j)}) / (|E(v,u)| - 1) \quad (3)$$

Here u, v are two users in a graph and B is a katz factor which is generally 0.05 and $\sigma(i, j)$ is the probability of having link type i given j. |E(u, v)| is the number of links between u,v. The score(v) is calculated by doing a BFS on the graph choosing a random node and then calculating the score between the parent and the child using the formula

$$p_{v,u} = 1 - (1 - \frac{1}{degree(u)})^{weight(u,v)}$$

Using this influence propagation formula we propagate influence between the two networks. After calculating flow values the top 10 flow values are chosen as neighbors. These neighbors are given as the input to the adsorption algorithm and the same procedure described in section 4.1 is repeated again for both ingredient and the recipe networks.

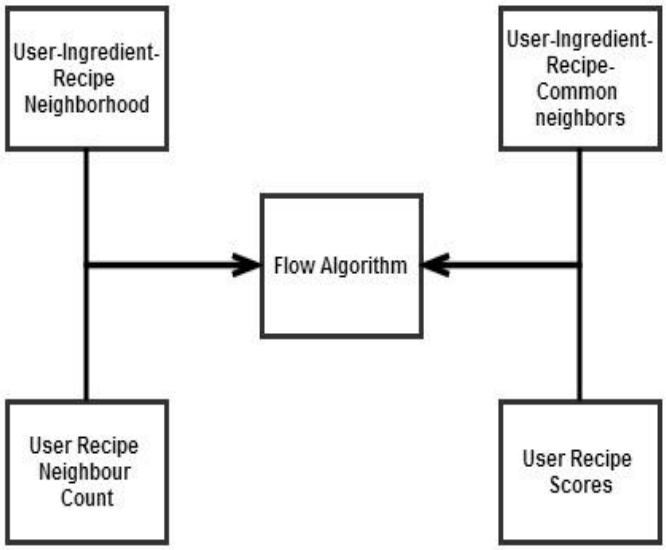


Figure 4.7 Workflow for calculating flow values

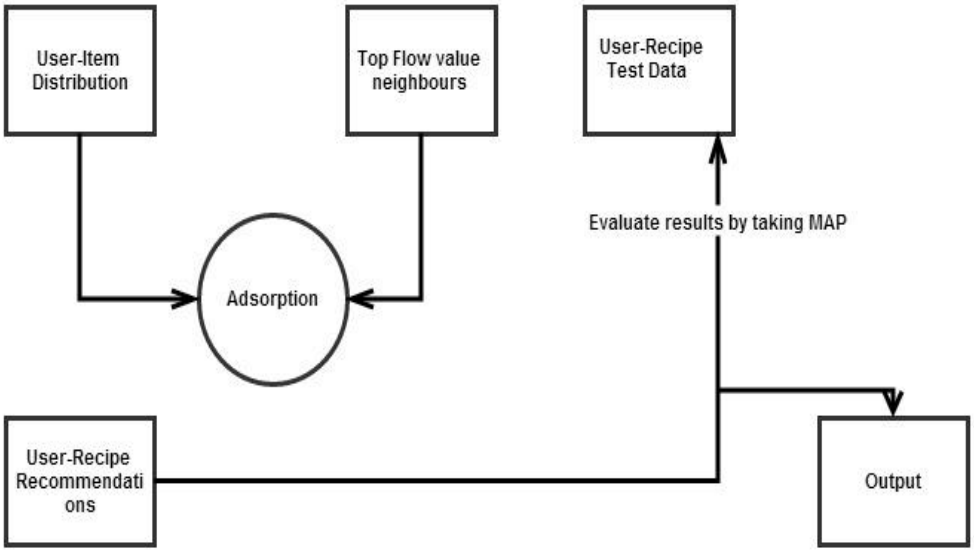


Figure 4.8 Adsorption workflow for user-recipe network with flow values

4.4 Mean Average Precision

Among all evaluation measures MAP has shown to have good discrimination and stability. The mean average precision is calculated by taking the average of precision of all queries[7].

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

A sample example to calculate mean average precision is given below:

i) If the user followed 1 and 3 and another node that wasn't recommended by the algorithm then the MAP is (1/1+2/3)/3=0.56

ii) If the user follows 1 and 2 and another node that wasn't recommended by the algorithm then the MAP is (1/1+2/2)/3 =0.67

iii) If the user follows 1 and 3 and no other node and 1 and 3 were recommended then the MAP is (1/1+2/3)/2=0.83

Precision and Recall (Algorithm Correctness)

To understand MAP well one needs to understand the what precision means in terms of information retrieval system. Precision and recall are the basic measures used in evaluating any kind of search strategy. It may involve searching for a document from a set of documents, recommending something to the user, searching for documents that involve some text etc.

We make certain assumptions in calculating the precision and recall. The assumption is that we already know what are the relevant documents in the data set given to us. In our case when we divide the data in to train and test folds **we know for sure** that the user likes 'x' and 'y' recipes and 'x' and 'y' ingredients. This is because we extracted this data from the input dataset. Such recipes or ingredients are considered to be relevant and other recipes that are not associated with the user are considered irrelevant. But, there exists another problem that there might exists some recipe that can occur in train as well as in recommendations. Such recommendations are biased and we filter those recipes so that recipes that are not in the train set are taken and evaluated. This is called **over sampling**.

When we know the set of relevant recipes, we see whether the recommendations made contain these relevant recipes or not. If the recommended set contains those recipes in which position did we get the recommendation. For example for user 'A' recommendations 'R1','R2' and 'R3' are made, then the position at which the recommendations were made are 1,2 and 3 respectively. Precision is the number of relevant documents retrieved to the total number of relevant and irrelevant documents.

C: Number of irrelevant documents retrieved

A: Number of relevant documents retrieved, then

$$\textbf{\textit{Precision = (A/(A+C))*100\%}}$$

MAP is the average of precision points in the recommendations made. MAP value for an algorithm indicates the average precision with which the algorithm recommended recipes to the user. The results section only contains the MAP values for the algorithm run on various folds. It indicates the average precision of the algorithm for that data set. MAP is the evaluation criteria used for all algorithms described in this paper.

Chapter 5 - Results

The results of all the algorithm are explained and tabulated in this chapter. The tables start from the individual networks i.e. user-recipe and user-ingredient networks and move to the entire network (user-recipe-ingredient) and results are shown for the three algorithms described in chapter 4. Firstly there are tables that explain how many users, items are there in each fold and how many edges are there in each fold. There tables give insights about how large the data is.

5.1 Statistics of different networks

The input data is divided into five folds and each fold is randomly divided and we generate recommendations for each fold. Each fold's recommendations are evaluated against their corresponding test recommendations and results are shown in tables below.

Table 5.1 Statistics of user-recipe network for all train folds

Fold	Users	Items	User-Item Count
1	15376	53793	1589017
2	15376	53754	1589017
3	15376	53832	1589017
4	15376	53778	1589017
5	15376	53777	1589017

These statistics show that in the train data of the user-recipe network, there are 15376 unique user vertices and approximately 53,770 (average of five folds) unique recipe vertices. There are about 1.5 million edges between user and recipe vertices in this graph. This graph is sent as input to the different algorithms and results are obtained.

Table 5.2 Statistics of user-recipe network for all test folds

Fold	Users	Items	User-Item Interaction
1	15376	46588	680747
2	15376	46624	680747
3	15376	46587	680747
4	15376	46573	680747
5	15376	46658	680747

In the test data there are 15376 unique user vertices and 46500(average of 5 folds) unique user vertices. There are about 680,700 user-recipe edges. This is graph which is compared against the recommended graph.

For the user-ingredient network the test set is same as the user-recipe test set. It is because when the algorithm recommends ingredients we calculate cosine similarity between recommended ingredients and all recipes and only top ten recipes are taken and evaluated against the user recipe test set. The training data though consists of all those ingredients of recipes for which user gave the rating greater than or equal to 4.The statistics of the user-ingredient network are shown below

Table 5.3 Statistics for user-ingredient network

Fold	Users	Items	User-Item Interaction
1	13868	8240	557050
2	13868	8280	555803
3	13868	8259	557752
4	13868	8242	557767
5	13868	8232	556125

The above table shows that in the user-ingredient network there are 13868 unique user vertices and 8250(average of five folds) unique ingredient vertices. There are about 550,700 (average of five folds) user-ingredient edges in the input graph. Though there are more ingredients for one recipe we only take the ingredients of those recipes for whom user rated 4 or 5. Since only these are taken into consideration we have relatively fewer edges in the graph.

5.2 Homogeneous Network Results

The results for the user-recipe homogeneous network for both recipe and ingredient network after running the adsorption algorithm are given below:

Table 5.4 Adsorption results for user-recipe network

Fold	Adsorption
1	0.008905858648788604
2	0.00985603092934859
3	0.007987452436075213
4	0.012890267076427202
5	0.005056082104359651
Average of all folds	0.008939138238999852

Table 5.5 Matrix factorization results for user-recipe network

Fold	A=1,L=0.01	A=1,L=0.1	A=5,L=0.01	A=5,L=0.1
1	3.249498290471236E-5	8.852179442710141E-6	1.3600870455709165E-5	8.155360652759195E-6
2	2.472158218125960 4E-5	7.316597294484911E-5	1.695076226814E-5	8.671522719389524E-6
3	5.6865574880663335E-5	5.3175016104256476E-5	5.428476454751169E-5	3.7883714880332987E-5
4	4.10426267281106E-5	4.7484329319657097E-5	5.0860029235419455E-5	4.3574401664932366E-5
5	5.16368531456981E-5	3.747336603736187E-5	6.915539368713145E-5	4.841600184992484E-5
Average of all folds	4.14E-05	4.40302E-05	4.09704E-05	2.93402E-05

Fold 1	Average = 5.593648316072873E-5
Fold 2	Average = 5.580228102340485E-5
Fold 3	Average = 9.006253819599293E-5
Fold 4	Average = 7.076065771435179E-5
Fold 5	Average = 4.352610364298338E-5
Average	6.32E-05

Table 5.6 Adsorption results for user-ingredient network

Table 5.7 Matrix factorization results for user-ingredient network

A--Alpha L-Lambda

Fold	A=1,L=0.01	A=1,L=0.1	A=5,L=0.01	A=5,L=0.1
1	2.0855270221165124E-4	9.267689906347549E-5	1.3029242473668853E-4	1.2480798771121354E-4
2	1.3566158556728276E-4	1.3196328435987644E-4	2.030065408057084E-4	1.41751007548354E-4
3	1.945027707579737E-4	1.8218714384817398E-4	1.5479700378243563E-4	1.4415632277885144E-4
4	1.0918118114398032E-4	1.0406859587400687E-4	1.0805852864905936E-4	1.2216265711973307E-4
5	1.3526026956047767E-4	1.3264848950332822E-4	1.388217878202269E-4	1.1428602398295428E-4
Average of all folds	1.57E-04	1.29E-04	1.47E-04	1.29E-04

5.3 Heterogeneous Network Results

The heterogeneous algorithm involves calculating flow values for both networks and taking neighbors having top flow values. The top ten flow neighbors are picked and the adsorption algorithm is given this new neighborhood. The top results are taken and MAP is calculated with the test data. The result are shown for user-recipe and user-ingredient networks.

Table 5.8 Heterogeneous algorithm results for user-ingredient network

Fold	Adsorption
1	7.055161207736649E-5
2	1.2226847034339227E-4
3	1.5518154452207522E-4
4	1.9133869728952968E-4
5	1.4363757990188788E-4
Average	1.37E-04

Table 5.9 Heterogeneous algorithm results for user-recipe network

Fold	Adsorption
1	0.007195661812678549
2	0.00837349680497323
3	0.009419522603670853
4	0.009135502654760807
5	0.008429704467803766
Average of all f[olds	0.008510778

5.4 Inferences

The heterogeneous algorithm is comparable to the matrix factorization in the user-ingredient network and better than the adsorption algorithm. For the user-recipe network the heterogeneous algorithm is way better than the matrix factorization and comparable with the adsorption algorithm. We can say that this algorithm is comparable or better in some cases compared to the matrix factorization and the adsorption algorithm.

5.3.1 User-Recipe Network

The adsorption algorithm for the user-recipe network yielded MAP value of 0.0089. The same adsorption algorithm while taking the top flow value neighbors yielded 0.0085. This is almost same as adsorption for homogeneous network. Since the data set is sparse the MAP values for both the algorithms are almost similar. For a denser data set the algorithm would perform even better. The top flow vales are selected by using a max priority queue. The priority queue filters the incoming flow values and arranges them in descending order.

When compared to matrix factorization, adsorption performs way better in homogeneous and heterogeneous setting. In homogeneous setting for the user-recipe network matrix factorization yields a MAP of 3.89352E-05. This value is way less than that of adsorption in both homogeneous and heterogeneous setting. For the user-recipe network we can see that adsorption performed way better than matrix factorization.

5.3.2 User-Ingredient Network

For the user-ingredient network the workflow of computing the results is completely different compared to the user-recipe network. The workflow is explained in detail in chapter 4. The algorithm recommends ingredients to user. But we are trying to recommend recipes for user. So we calculate cosine similarity between the ingredients and all recipes in the network. The top ten cosine similarity recipes are assumed to be correct. The cosine similarity is a most common metric used in information retrieval systems to calculate the similarity between two entities. The procedure for calculating cosine similarity is given below,

1) Assume for a user A the algorithm recommends ingredients I1,I2,I3. Assume there are 4 recipes in a system and each recipe has the following ingredients
    a) R1: I1,I2
    b) R2: I1,I2,I3,I4

c) R3: I1

d) R4: I1,I2,I3

2) We calculate cosine similarity using the following formula: cos(theta) = $\Sigma w_{IjRj} * w_{IjU} / (|Rj||U|)$ (where j ranges from 1..4)

3) Here the weights are usually taken as 1 since we only want to see if the ingredient appears in a recipe or not. The same is done for all the recipes and MAP is taken between the recipes that we calculated using cosine similarity and test recipes. The results are shown in the above tables.

The average of adsorption results for homogeneous network (user-ingredient) for the five folds is 6.32E-05 where as it is 1.37E-04 for the heterogeneous network. Since ingredients and recipes are interrelated taking neighbors with high flow values in this case is better than the homogeneous network. The matrix factorization average for different parameters is 0.0001405 which is slightly better than that of heterogeneous network. Hence in this case the heterogeneous algorithm is comparable to matrix factorization. Below is a table reflecting this comparison

Type of Network	Adsorption (Homogeneous)	Matrix Factorization (Homogeneous)	Adsorption (Heterogeneous)
Ingredient	6.32E-05	0.0001405	1.37E-04
Recipe	0.0089	0.0085	3.89352E-05

Table 5.10 Comparison for all networks

5.5 Testing

Unit Testing

Unit testing is the method by which individual components of the system are tested This project involves a lot of string parsing to be done. Data cleaning is the first step where a lot of string parsing is done. The different cases for which I have unit tests are:

a) I have written test cases to check if two similar strings, after cleaning return the same output or not. For example the two strings pinch of salt and salt after cleaning should give salt.

b) I have written test cases to compute weights and degrees of one node. After testing it for some inputs I tested it on the input data set.
c) I have written test cases to compute MAP between the recommends results and the test set for one user.
d) I have written test cases to compute flow value of a node in one network and tested it using other data of similar kind. Both returned the same value and hence I deduced that the algorithm is computing the flow values correctly.
e) I computing the K nearest neighbors we use a max priority queue. I have used the Java version of priority queue which is a min priority queue and converted the same to max and tested it on one line of the input and it worked correctly.

Performance Testing

I have mainly used hadoop map reduce to write all my code hence performance is also a key factor in deciding the scalability of an algorithm. For example consider the statistics shown in Table 5.2. The number of user item interactions are 1589017. Before I talk about how performance and running time reduces greatly because of using hadoop I will talk about another technology called Titan. Titan is a graph database that uses either Hbase or Cassandra as it's back end. Titan stores graph in a form that is readable only by the software. It uses indexing and other methods to actually index keys in the graph for efficient and fast retrieval.

The advantages of Titan over other graph database are:

a) We can easily plug in Titan to Hadoop and use it for efficient retrieval process.
b) Indexing of nodes and labels in the graph allows faster retrieval when compared to other graph databases.
c) Applications can be built on top of Titan.
d) It also provides an application server that runs Titan applications in background.

While designing the workflows of this application I build graphs using Titan but there are two ways to build a graph using Titan standalone mode and distributed mode. Using hadoop we can create Titan graph database. I tried both the variants. The standalone mode of Titan took me about 1-1.30 Hrs to create the Graph where as the distributed mode took me about 10-20 min when 30 nodes were allocated.

I decided not to use Titan and go ahead with standard text files because of the following reasons:

a) Titan comes with other modules like an application server and application module which is not necessary for this project.
b) Titan uses Hbase to store the graph and every time a query is issued to the graph Titan connects to the database builds the graph in memory and displays the results. Since database calls are expensive and the project that we intended to build does not require an exclusive graph database.
c) Complexity of the project increases and such complexity is not required since the project does not require complex graph retrievals.

Because of the above mentioned reasons the idea of using Titan was dropped and we went ahead in our research using text files as input. When we require the recommendation system to be built as an application Titan comes in handy.

When we used the standard text files for processing the process of building the input only took 2 minutes approximately. So when large number of nodes are allocated for processing the input the process of building the graph becomes extremely quick. Since map-reduce is a framework build for distributed environment the programs make use of distributed memory to accomplish the task faster

Chapter 6 - Conclusion and Future Work

6.1 Conclusion

The heterogeneous algorithm is bound to give better results because it learns from other networks tries to pass on that knowledge each individual network. Since adsorption has proven to give better results(better than existing algorithms) in homogenous network, the 'flow' information from other networks increases the accuracy of the adsorption algorithm.

The results reflect the same in the results shown in chapter 5. Since the recipe data-set is sparse, meaning a recipe 'A' has few ingredients like salt, pepper which are found in almost all recipes and there are some ingredients only used in one or two recipes, the results do not improve that much but they are comparable to that of homogeneous network. This is a great sign because, using knowledge of other networks, it is able to give comparable results to that of homogeneous network. For a more denser data set the results would be even better.

6.2 Future Work

The following things can be done as a part of future work to this project:

1. The heterogeneous algorithm described in this paper can be extended to other data sets to test its effectiveness.
2. It can be tried on denser data sets to test whether it yields better results or not.
3. There are several variants to the adsorption algorithm. These variants can use the flow data and we can see which variant of adsorption goes well with the flow data.
4. The flow values can be plugged in to other algorithms as well other than adsorption to check if they work well with other algorithms or not.

References Or Bibliography

[1] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly.
Video suggestion and discovery for youtube: taking random walks through the view graph.
In WWW, 2008.

[2] Partha Pratim Talukdar and Koby Crammer,New Regularized Algorithms for Transductive Learning
Kaggle, World's largest community of data scientists, world wide web

[3] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the spreadof influence through a social network. In Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 137146, 2003.

[4] R. Singh, J. Xu, and B. Berger. (2008) Global alignment of multiple protein interaction networks with application to functional orthology detection, Proc. Natl. Acad. Sci. USA.

[5] L. Tang and H. Liu. Relational learning via latent social dimensions. In KDD ’09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY,USA, 2009

[6] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation.
Technical report, CMU CALD tech report, 2002

[7] Predicting Links in Multi-Relational and Heterogeneous Networks,Yang Yang and Nitesh Chawla, Yizhou Sun and Jiawei Hanı, ASONAM '11 Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining

Appendix A - Technologies

The major technologies used in my research are:

1. Apache Hadoop
2. Mahout
3. Pig
4. Apache Http Client
5. Google Gson Api

1. Apache Hadoop

Apache Hadoop is an open source software used to process huge amounts of data using distributed computing. As data is exploding day by day traditional file processing techniques become irrelevant when it comes processing huge data. In a distributed computing environment the nodes are prone to failures. Instead of relying on hardware to provide relentless service apache hadoop manages the nodes in the cluster, failure management, delivering highly available service on top of a cluster.

Most of the technologies I have used in this project are Apache Hadoop related projects. The apache hadoop project includes Hadoop Common, HDFS, YARN and MapReduce[1]. I have mostly written MapReduce programs for my research.

1.1 MapReduce

MapReduce programming model is used to process large data sets using distributed computing. The MapReduce programs usually have a map() task and a reduce() task. The names were inspired by functions used in functional programming although the task performed differs from its original form[2]. The steps taken in a MapReduce program to take the input and generate output is as follows.

1. The "MapReduce" framework divides the input data into chunks(64 Mb each) and sends it to different nodes in the cluster. Every 64 Mb chunk is called a "Map Task:", that is processed by a node.

2. The framework then calls the map method on each key value pair (K1) in the task and the output of this is also a key value pair K2.
3. All the intermediate key-value pairs (K2) are sorted and all values belonging to key K2 are aggregated.
4. The reduce operation is then called on the key-value pair K2 which now contains a key and a list of values to process.
5. The reducer then emits an output key value pair K3 which is the final result.

-

2 Mahout

Mahout is a framework that provides implementations of scalable machine algorithms focused on collaborative filtering. The algorithm that I have used in Mahout framework for my project is matrix factorization. Mahout can be downloaded from the Apache Mahout's home page (https://mahout.apache.org/)

3 Pig

Pig is a high level tool for creating MapReduce programs using Hadoop. The language in which programs are written are called Pig Latin. Pig Latin abstracts the Java implementations of MapReduce to provide a SQL kind of tool for writing MapReduce programs. Users can also write user defined functions in Python, Groovy, Java to have their own implementations of MapReduce used. I have written Pig Latin queries to evaluate the results of the algorithm.

4 Apache Http Client

Apache Http Client is a Java API that allows the user to make raw http requests to a web server. Usually REST API's in Java are built using this API Using this API one can control number of connections made to the web server, parameters being sent along with the request to the server and also specify how many requests to make per second. I used the Apache Http Client to send requests to allrecipes.com website and downloaded the response and parsed through the HTML file to get the data that I want.

5 Google GSON API

Google GSON API is used to convert model objects to JSON strings and vice versa. The data that I crawled from the website of allrecipes.com is parsed and stored in model objects. These model objects are converted to JSON strings which are stored in a text file. The text file is the input file that is sent to the algorithm for processing.

The other API's that I used are JSOUP API which is used to parse XML documents. The API comes with a SAX parser but the advantage of using this API's over other API's (that parse XML documents) is that JSOUP provides DOM (Document Object Model) methods in Java Script in its library. So if we do know how to parse a DOM tree in Java Script then using this API is pretty simple.

I have used beocat cluster provided by Kansas State University, CIS department to run all my jobs. Since I worked with beocat and submitted bunch of jobs, I had to write a lot of shell scripts. I used Eclipse IDE to code my project and Java was the only object oriented language that I used.